[CMSC 425/525] Assignment #2: Graph Coverage

Dr. Kosta Damevski
Due: End of Day, February 26th, 2023

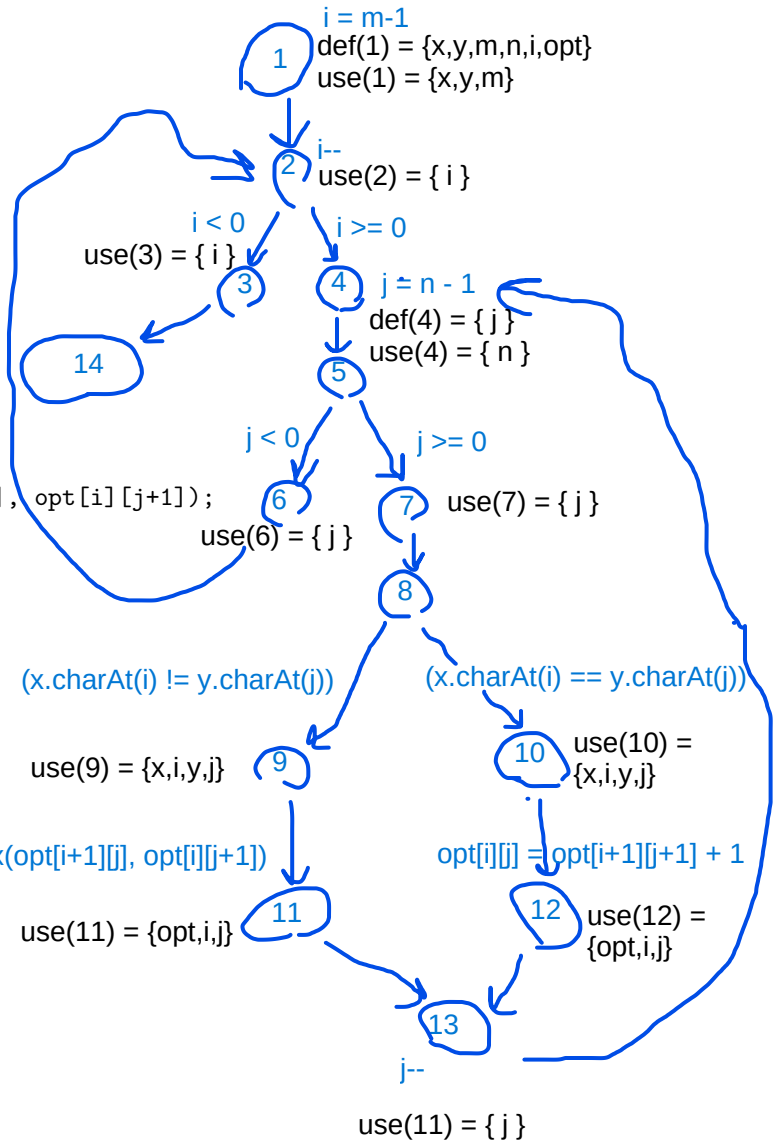
Given the following implementation of the longest common subsequence algorithm:

```
/*
 * find the longest common subsequence between two strings. Note that
 * unlike substrings, subsequences are not required to occupy consecutive
 * positions within the original sequences.
 * @param x - the first string
 * @param y - the second string
 * @returns the longest subsequence
 *
 * example: GGCACCACG,ACGGCGGATACG => GGCAACG
 */
public static String lcs(String x, String y) {
    int m = x.length(), n = y.length();
    int[][] opt = new int[m+1][n+1];
    for (int i = m-1; i >= 0; i--) {
        for (int j = n-1; j >= 0; j--) {
            if (x.charAt(i) == y.charAt(j)) {
                opt[i][j] = opt[i+1][j+1] + 1;
            }
            else {
                opt[i][j] = Math.max(opt[i+1][j], opt[i][j+1]);
            }
        }
    }

    // Recover LCS itself.
    String lcs = "";
    int i = 0, j = 0;
    while (i < m && j < n) {
        if (x.charAt(i) == y.charAt(j)) {
            lcs += x.charAt(i);
            i++;
            j++;
        }
        else if (opt[i+1][j] >= opt[i][j+1]) {
         i++;
        }
        else {
         j++;
     }
```

1 - 14

14 - 21 below



```
    }
    return lcs;
}
```

Complete the following tasks:

1. Draw a control flow graph for the `lcs` method. Enumerate the nodes in the graph.
2. Label the control graph with defs and uses at the appropriate nodes.
3. Compute the *all-def-use* paths criteria for the graph.
4. Implement a set of JUnit tests that cover as many of the *all-def-use* paths (test requirements) as possible.
5. Reflecting on the JUnit test you have written in the previous task, are there any tests that you would have written if you only used your intuition and did not follow the *all-def-use* criteria. Are there any tests that you ended up writing that you would not have otherwise thought of?

Turn in your code and associated graphs as an additional document in a format I can easily open (e.g. MS Word, PDF).

all-def-use

int i:
[1,2,3,12]
[1,2,4,5,7,8,9,11]
[1,2,4,5,7,8,10,12]
[1,2,4,5,7,8,9,11,13,4,5,6,2,3,14]
[1,2,4,5,7,8,10,12,13,4,5,6,2,3,14]
[14,15,16]
[14,15,17,18]
[14,15,17,19]

int j:
[4,5,6]
[4,5,7,8,9,112]
[4,5,7,8,10,12]
[14,15,16]
[14,15,17,18]
[14,15,17,19]

String x, String y:
[1,2,4,5,7,8,9]
[1,2,4,5,7,8,10]

int [] [] opt:
[1,2,4,5,7,8,9,11,13,4,5,6,2,3,14,15,17,19]
[1,2,4,5,7,8,10,12,13,4,5,6,2,3,14,15,17,19]